Analiza zależności kosztu komórki od parametrów symulacji w grze Cell Evolution

Mikołaj Kubś

Czerwiec 2024

Spis treści

1 Wstęp

Poniższa praca będzie zajmować się badaniem zależności średniego kosztu komórki (average creature cost) w grze Cell Evolution od początkowych parametrów symulacji. Koszt komórki oznacza ilość energii, jaką organizm potrzebuje, by przeprowadzić mitozę ("koszt"genu/części ciała/akcji w raporcie będzie zawsze oznaczał koszt energetyczny - czyli jedzenie przetworzone na energię). Parametrami początkowymi będą:

- wielkość mapy (map size),
- liczba początkowych komórek (starting creatures),
- okres czasu między powstaniem nowego jedzenia (time to spawn a food),
- mnożnik mutacji (mutation range),
- mnożnik prawdopodobieństwa mutacji części ciała (body part mutation chance),
- okres czasu między reprodukcją (interval between births),
- mnożnik wpływu wieku na głód (hunger mult from age),
- mnożnik agresywności (aggresivnes mult),
- mnożnik kosztu genów (genes cost mult),
- mnożnik kosztu części ciała (body parts cost mult).

Najpierw zostanie przeprowadzona regresja liniowa, by estymować koszt komórki z parametrów początkowych. Następnie sprawdzona będzie hipoteza, czy uwzględnienie w modelu innych danych końcowych (rezultatów) symulacji zwiększy jego jakość.

Dodatkowo zostaną utworzone trzy modele regresji, które będą ze sobą porównane po zoptymalizowaniu.

Pomysł projektu zainspirowany był chęcią sprawdzenia, czy w grze Cell Evolution dałoby się przewidzieć różne dane końcowe na podstawie parametrów symulacji. Oznaczałoby to, że różne symulacje o tych samych parametrach kończyłyby się podobnym skutkiem. W grze występuje jednak dużo losowości - jedzenie powstaje w różnych miejscach, agenci przy polowaniu wybierają swoje ofiary niederministycznie, mutacje odbywają się losowo i wiele więcej.

2 Opis działania symulacji w Cell Evolution

W tej grze, na podstawie parametrów początkowych, tworzona jest mapa wraz z początkową populacją, której wartości wszystkich genów są bliskie 0,5. Następnie komórki te konkurują między sobą o ograniczone zasoby jedzenia, starając się zdobyć energię potrzebną do reprodukcji. Mogą ją zdobyć na 2 sposoby - zjeść plankton, który powstaje w losowych miejscach na mapie lub zjeść ciało innej komórki, która zmarła w sposób naturalny czy w wyniku polowania. Wraz z wiekiem wzrasta koszt utrzymania życia. Gdy komórki zdobędą wystarczającą ilość

energii i minie odpowiednia ilość czasu między mitozami, mogą przeprowadzić reprodukcję. Jest to proces długotrwały, który je spowalnia. Po jego zakończeniu powstaje nowa komórka, która dziedziczy po jedynym rodzicu geny i części ciała. Na podstawie parametrów początkowych przeprowadzona jest jej mutacja, zmieniająca wartość każdego genu. Nowa komórka potencjalnie tworzy nową część ciała w losowym miejscu lub usuwa istniejącą. Komórka może mieć od 0 do 5 części ciała. Kolce obronne i ofensywne służą do walki, kompostownik do przetwarzania odpadów na energię, a turbiny do przyspieszenia ruchu.

Koszt komórki to matematyczna funkcja, która jest obliczona z wartości genów i kosztu poszczególnych części ciała. Jest to inna wartość dla każdego organizmu.

3 Zbiór danych

3.1 Źródła danych

Mając dostęp do kodu źródłowego gry, możliwe było napisanie skryptu, który automatycznie zapisywał dane z symulacji co określony czas (500 symulowanych sekund). Dodatkowo każda próba była inicjowana automatycznie, z parametrami początkowymi ustawionymi losowo. Wszystkie symulacje były przyspieszone 16-krotnie za pomocą wbudowanej do gry możliwości.

3.2 Tworzenie danych

Oto parametry, które były losowo dobierane przy tworzeniu nowej symulacji:

- wielkość mapy (map size),
- liczba początkowych komórek (starting creatures),
- okresu czasu między powstaniem nowego jedzenia (time to spawn a food),
- mnożnika mutacji (mutation range),
- mnożnika prawdopodobieństwa mutacji części ciała (body part mutation chance),
- okresu czasu między reprodukcją (interval between births),
- mnożnika zależności głodu od wieku (hunger mult from age),
- mnożnika agresywności (aggresivnes mult),
- mnożnika kosztu genów (genes cost mult),
- mnożnika kosztu części ciała (body parts cost mult).

Żeby utrzymać dane bliżej standardowych, domyślnych wartości, zastosowana była następująca metoda generowania wartości losowych:

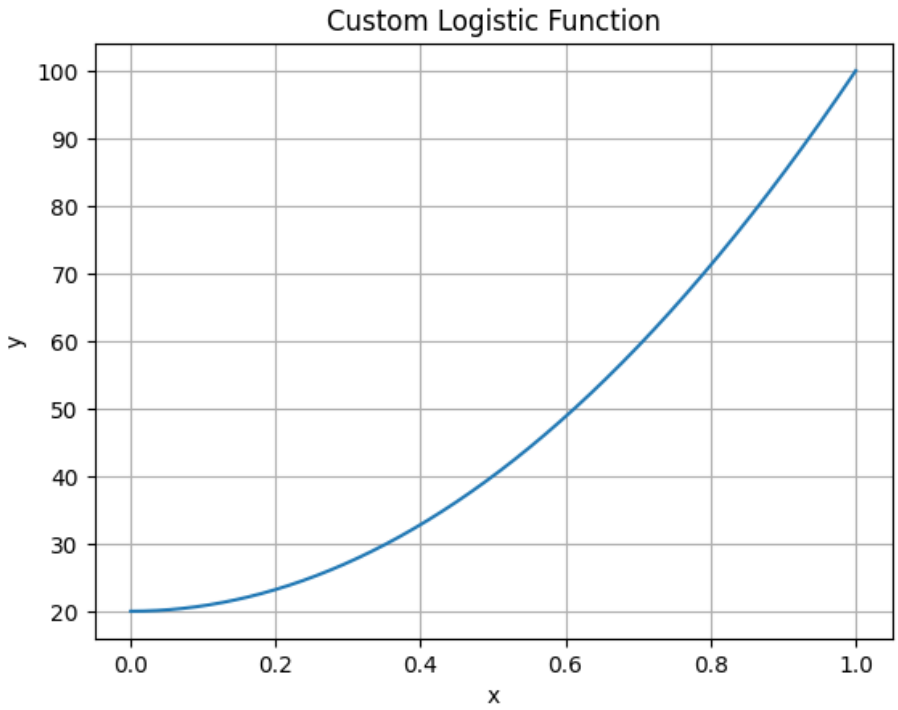
$$m_{\text{inter}} = \frac{\text{mean} - \text{min}}{\text{max} - \text{min}}$$
$$\text{custom\_logistic}(x) = x^{\log_{0.5} m_{\text{inter}}} \cdot (\text{max} - \text{min}) + \text{min}$$

Oto funkcja wyrażona w kodzie C# używanym w silniku gry Unity, w którym zbudowano grę.

```
float GenerateRandomValue(float min, float max, float mean)
{
    float meanWeight = meanWeightMultiplier * Random.value;

    float x = (Random.value + .5f * meanWeight) / (1 + meanWeight);

    float value = CustomPowerFunction(x, min, max, mean);

    return value;
}

float CustomPowerFunction(float x, float min, float max, float mean)
{
    float meanInterpolated = (mean - min) / (max - min);

    return Mathf.Pow(x, Mathf.Log(meanInterpolated, 0.5f)) * (max - min) + min;
}
```

Rysunek 1: Funkcja CustomPowerFunction, oraz dodatkowa funkcja, która przybliża x do 0.5 na podstawie dodatkowego parametru wagi mediany.

Metoda ta pozwala na generowanie wartości losowej dla podanej wartości minimalnej i maksymalnej, a także wybranej mediany. Oznacza to więc, że połowa danych będzie poniżej mediany, a druga połowa powyżej.



Rysunek 2: Rozkład losowanych wartości dla minimum = 20, mediana = 40, maksimum = 100. Podane przykładowe dane są faktycznie używane w grze do ustalenia wielkości mapy.

Na początku każdej symulacji tworzony jest plik "Initial settings x - *data i godzina*.csv". Zawarty jest w nim szereg informacji - są to wszystkie parametry, na podstawie których stworzona jest mapa, ustawienia oraz początkowe komórki.

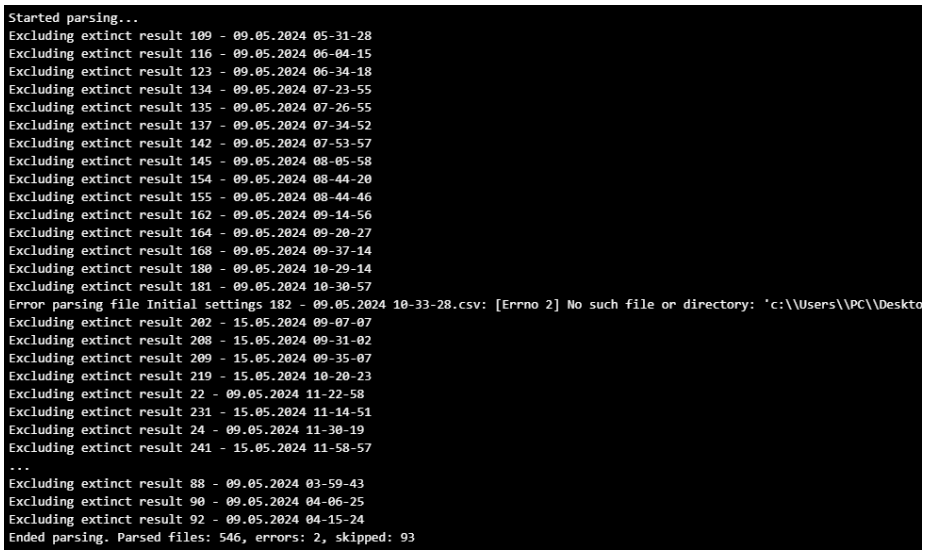
```
PlayerPrefs.SetFloat("mapSize", GenerateRandomValue(20, 100, 40));

PlayerPrefs.SetFloat("startingCreatures", GenerateRandomValue(1, 60, 25));

PlayerPrefs.SetFloat("timeToSpawnAFood", 1f / GenerateRandomValue(0.1f, 15, 4));

PlayerPrefs.SetFloat("mutationRange", GenerateRandomValue(0.01f, 1f, 0.1f));

PlayerPrefs.SetFloat("bodyPartMutationChance", GenerateRandomValue(0.005f, 1f, 0.05f));

PlayerPrefs.SetFloat("intervalBetweenBirths", GenerateRandomValue(0, 100, 40));

PlayerPrefs.SetFloat("hungerMultFromAge", GenerateRandomValue(0, 3f / 300f, 1f / 300f));

PlayerPrefs.SetFloat("newCellCreationCostMult", 1);

PlayerPrefs.SetFloat("aggresivnesMult", GenerateRandomValue(0, 5, 1));

PlayerPrefs.SetFloat("genesCostMult", GenerateRandomValue(0, 5, 1));

PlayerPrefs.SetFloat("bodyPartsCostMult", GenerateRandomValue(0, 5, 1));
```

Rysunek 3: Parametry ustawiane przy nowych symulacjach, wraz z wartościami minimalnymi, maksymalnymi i domyślnymi.

Następnie zapisywane są dane pozyskane po uruchomieniu symulacji w plikach "Attempt x - *data i godzina*.csv. Co 500 symulowanych sekund zapisane są dane dotyczące ilości komórek, a także wiele średnich - kosztu komórki, genów, a także ilości i rodzaju części ciała. W tym raporcie nie podjęto się analizy zmiany tych wartości w czasie, a jedynie przeanalizowano ostatni wiersz - zapisany zaraz przed końcem symulacji. Taka analiza na pewno również mogłaby być interesująca.

Analizowanym zbiorem danych są więc krotki zawierające dla danej symulacji parametry początkowe oraz ostatni wiersz udokumentowanych rezulatów. Krotki te są formowane w pliku analyzer.py, łączącym pliki "Initial settings x..."i "Attempt x..."na podstawie numeru x w nazwie. Ich odczytem zajmuje się biblioteka *csv*.

Jedną z pierwszych decyzji było odrzucenie wszystkich symulacji, które zakończyły się przedwcześnie - z powodu wymarcia wszystkich komórek lub nadmiernego wzrostu populacji (ponad 200 komórek żyjących naraz). Decyzja ta była zmotywowana faktem, że dane te zbyt mocno różniłyby się od oczekiwanych i kończyłyby się wcześniej, niż cała reszta.



Rysunek 4: Z 641 wierszy 2 nie miały odpowiadającego im pliku "Attempt x...", a 93 pominięto z powodu przedwczesnego zakończenia. Wśród pominiętych danych, tylko 1 symulacja doświadczyła nadmiernego wzrostu populacji.

Parametr początkowy	**Wartość**
mapSize	23.761980
startingCreatures	15.000000
timeToSpawnAFood	4.529128
hungerMult	0.750000
speedMult	2.000000
mitosisSpeedMult	1.000000
mutationRange	0.383709
bodyPartMutationChance	0.005873
intervalBetweenBirths	64.644820
hungerMultFromAge	0.003437
newCellCreationCostMult	1.000000
aggresivnesMult	1.202730
genesCostMult	0.254597
bodyPartsCostMult	0.791882
initialSize	0.300000
fullyRandomiseStartingGenes	0
allCreaturesStartAsDesignedCreature	0
startAsACreature	0
colorfulCells	0
speedGeneOn	1
sensorGeneOn	1
processingGeneOn	1
gluttonyGeneOn	1
reproductionGeneOn	1
storageGeneOn	1
parentEmpathyGeneOn	1
aggresivnessGeneOn	1
healthGeneOn	1
foodPreferenceGeneOn	0
fightingPermitted	1

Tabela 1: Przykładowe wartości parametrów początkowych.

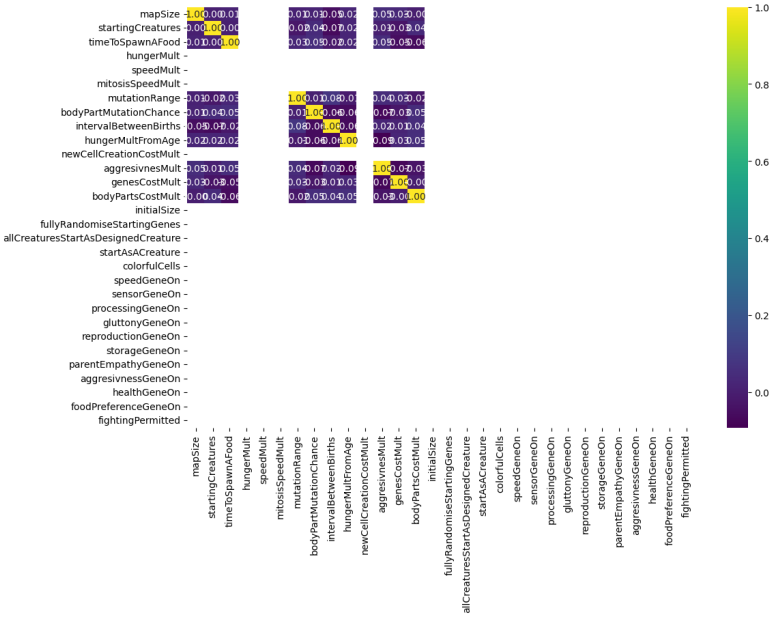
Rezultat	Wartość
simulation time	4500.051000
speed gene	0.991227
sensor gene	0.416732
processing gene	0.814775
gluttony gene	0.573346
reproduction wilness gene	0.307748
energy storage capacity gene	0.733310
parental empathy gene	0.345123
aggresivness gene	0.866806
health gene	0.861556
food preference gene	0.500000
creatures count	18
average creature cost	41.878194
average amount of body parts	0.000000
average amount of protective spikes	0.000000
average amount of composters	0.000000
average amount of turbines	0.000000
average amount of offensive spikes	0.000000

Tabela 2: Przykładowe wartości rezultatów.

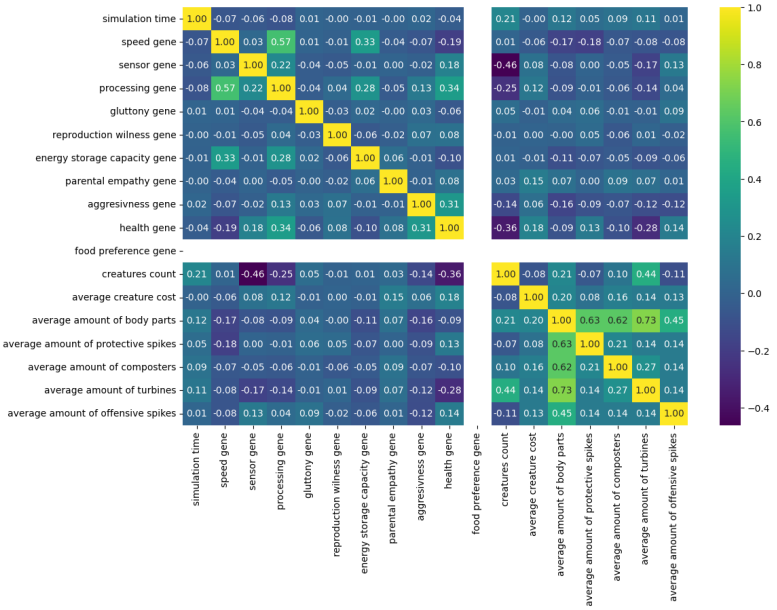
3.3 Oczyszczanie danych

Przed właściwą analizą danych, wymagają one wstępnego przetworzenia i oczyszczenia.

Wiele kolumn będzie można usunąć z powodu braku ich istotności. Najpierw warto przeanalizować macierze korelacji obu tabel.



Rysunek 5: Macierz korelacji parametrów początkowych przed ich oczyszczeniem.

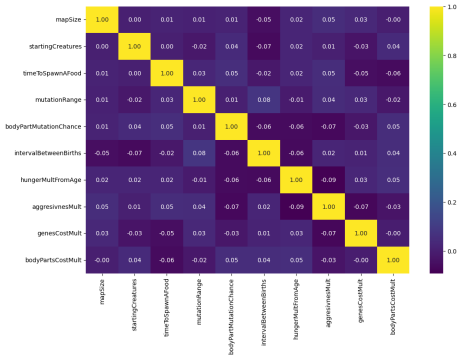


Rysunek 6: Macierz korelacji rezultatów przed ich oczyszczeniem.

Oczywistym wnioskiem jest usunięcie danych, które są stałe dla każdej symulacji. Są to białe wiersze/kolumny. Dodatkowo usunięta została kolumna "simulation time", ponieważ jej wartości zawsze będą zbliżone do 4500.

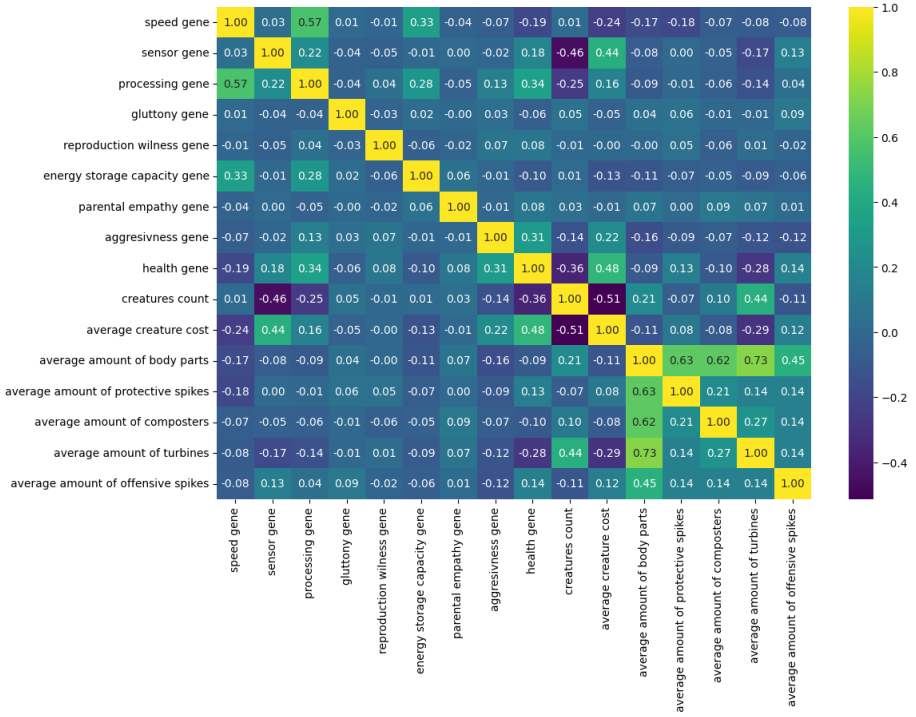
4 Analiza i dalsze przetwarzanie przygotowanych danych

4.1 Eksploracyjna analiza danych



Rysunek 7: Macierz korelacji parametrów początkowych.

Ponieważ dane były generowane losowo w sposób niezależny od siebie, korelacje parametrów początkowych w zależności od innych są bliskie 0. Był to oczekiwany rezultat.



Rysunek 8: Macierz korelacji rezultatów.

W tej macierzy korelacji można zaobserwować wiele interesujących zależności. Oto niektóre z nich:

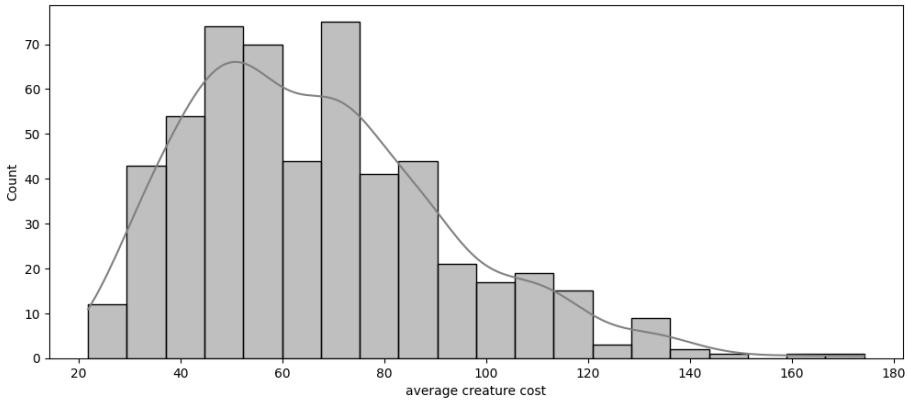
Niektóre z nich są oczywiste – średnia liczba części ciała jest oparta na średnich ilościach innych części ciała. Istnienie części ciała może oznaczać, że parametr początkowy odpowiadający za szansę mutacji części ciała jest wyższy, co sprawia, że istnienie innych części staje się bardziej prawdopodobne.

Występuje silna korelacja między genem przetwarzania jedzenia, a genem prędkości. Przy większej liczbie stworzeń, mniej korzystne jest posiadanie lepiej rozwiniętego genu sensorycznego.

Interesującą, silną korelacją jest ta między liczbą stworzeń a średnią liczbą turbin. Wydaje się, że przy większej liczbie stworzeń ważne jest, aby dotrzeć do jedzenia przed innymi. Z drugiej strony, ta sama korelacja nie jest obserwowana między liczbą stworzeń a genem prędkości. Jednym z powodów może być niezamierzony błąd w grze – dane były zbierane przy prędkości symulacji wynoszącej 16, co jest dość dużą wartością. Z tego powodu stworzenia o dużej prędkości mogą „mijać" jedzenie, ponieważ mają mniej klatek do dostosowania swojego kierunku ruchu. Turbiny znacznie zwiększają prędkość skręcania, podczas gdy gen prędkości tego nie robi. Możliwe więc jest to, że wartości te byłyby inne dla mniejszej prędkości symulacji. Innym wytłumaczeniem mogłoby być to, że turbiny energetycznie opłacają się bardziej niż gen szybkości.

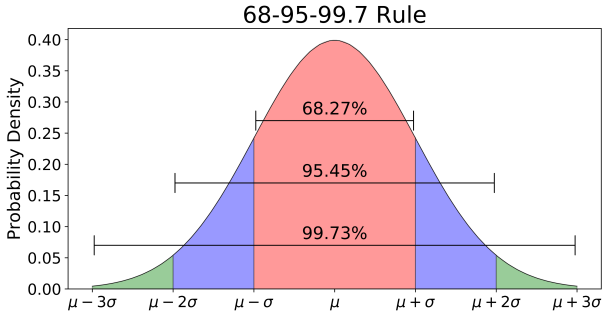
Jedyną daną z rezultatów, która pozostanie, jest średni koszt komórek. W następnym dziale przeprowadzona będzie estymacja tej kolumny nie tylko z parametrów początkowych, ale też z reszty rezultatów.

4.2 Usuwanie wartości odstających (outlierów)

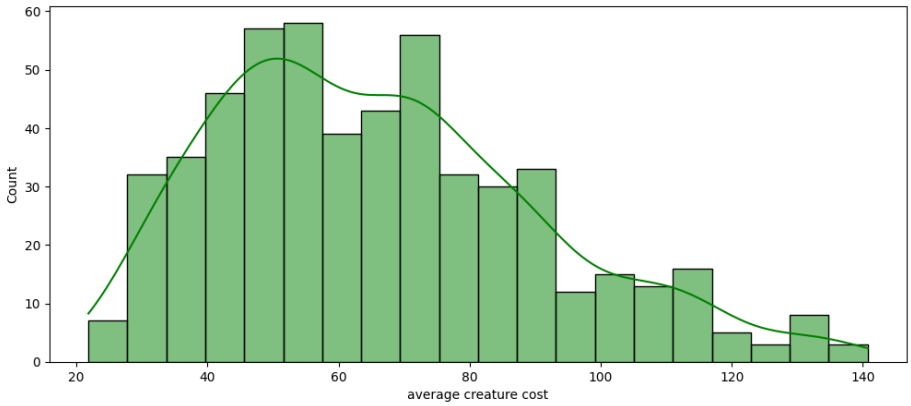


Rysunek 9: Histogram średnich kosztów komórek, wraz z wartościami odstającymi.

Przy pomocy reguły trzech sigm wiersze, dla których średni koszt komórek odstawał w sposób zbyt znaczący od normy, zostały usunięte. W praktyce oznacza to, że około 0.3% najbardziej odstających danych zostało usunięte, czyli te, których średni koszt komórek był większy od około 140.



Rysunek 10: Reguła trzech sigm



Rysunek 11: Histogram średnich kosztów komórek, bez wartości odstających.

4.3 Ogólna jakość danych

Ponieważ dane zostały wytworzone w sposób kontrolowany, nie ma wśród nich nieprawidłowości często spotykanych przy analizie danych pobranych z internetu. Nie występują zduplikowane ani fałszywe wartości. Jedynym problemem był dodatkowy znak ";"w nagłówkach plików .csv. Został on usunięty ze wszystkich plików.

4.4 Estymacja rezultatów z parametrów początkowych

Za pomocą modelu LinearRegression z modułu sklearn przeprowadzono estymację różnych rezultatów z parametrów początkowych. Podane wyniki to wskaźnik R^2.

Gene/Metric	**Estimation score**
Speed gene	0.2966
Sensor gene	0.2700
Processing gene	0.5678
Gluttony gene	0.0168
Reproduction wilness gene	-0.0006
Energy storage capacity gene	0.1768
Parental empathy gene	0.0567
Aggressiveness gene	0.0751
Health gene	0.4507
Creatures count	0.6871
Average creature cost	0.8060
Average amount of body parts	0.4945
Average amount of protective spikes	0.2272
Average amount of composters	0.2165
Average amount of turbines	0.3628
Average amount of offensive spikes	0.1533

Tabela 3: Tabela wyniku estymacji rezultatów z parametrów początkowych.

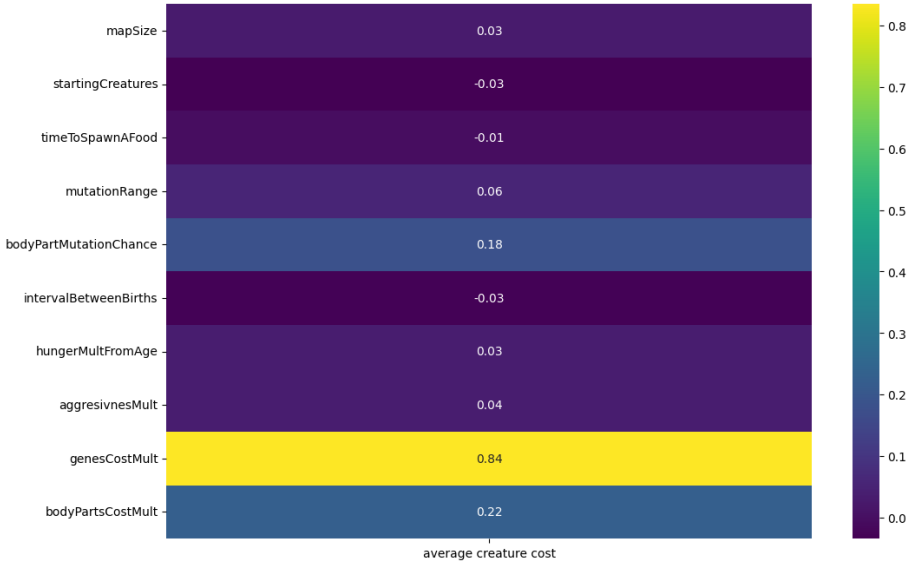
Z tych danych wyciągnięto następujące wnioski:

1. Najwyższy wynik regresji był dla średniego kosztu komórek - 0,8060.
2. Wyniki regresji niektórych innych rezultatów osiągnęły umiarkowanie wysokie wartości - w tym dla genu przetwarzania jedzenia, genu zdrowia czy też średniej ilości części ciała,
3. Większość regresji osiągnęła wynik niski lub bardzo niski.

Dane analizowane w raporcie dotyczą średniego kosztu komórki, który osiągnął wynik około 0,8. Wynik ten sugeruje, że zależność między tą daną a parametrami początkowymi jest dość silna.

5 Regresja liniowa na podstawie parametrów początkowych

5.1 Kolumna korelacji



Rysunek 12: Kolumna korelacji średniego kosztu komórki od parametrów początkowych symulacji.

Ponieważ wartość korelacji jest niska dla wielu kolumn, estymacja powstanie na bazie tylko trzech parametrów początkowych. Wybrane zostały:

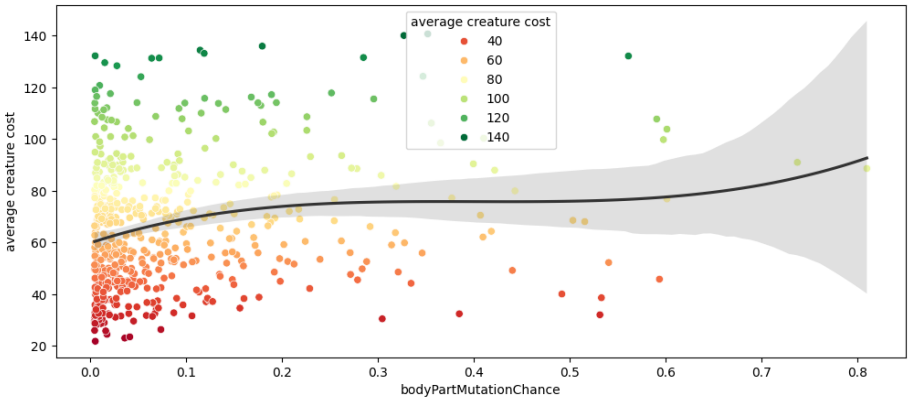
- mnożnik agresywności
- czas między powstaniem jedzenia
- mnożnik kosztu genów

5.2 Wynik regresji liniowej

Po przeskalowawaniu danych za pomocą StandardScaler, oto wyniki regresji:

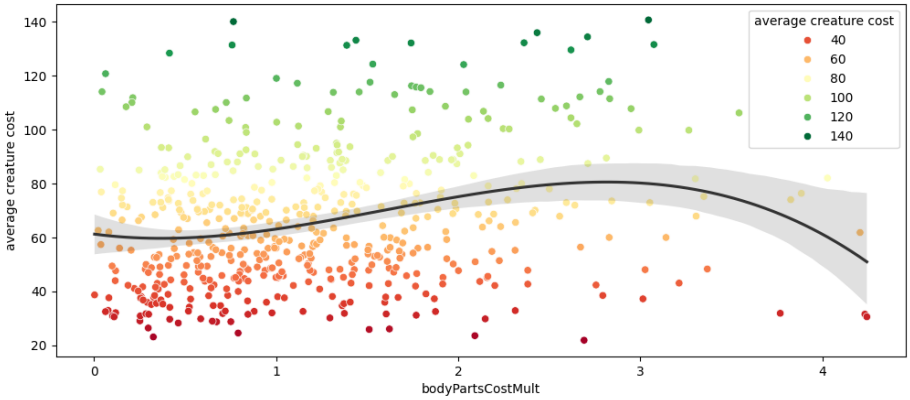
Data type	R^2
Test data	0.7631864002167397
Train data	0.7937762753894901

5.3 Wykresy wartości parametrów początkowych, a średniego kosztu komórki



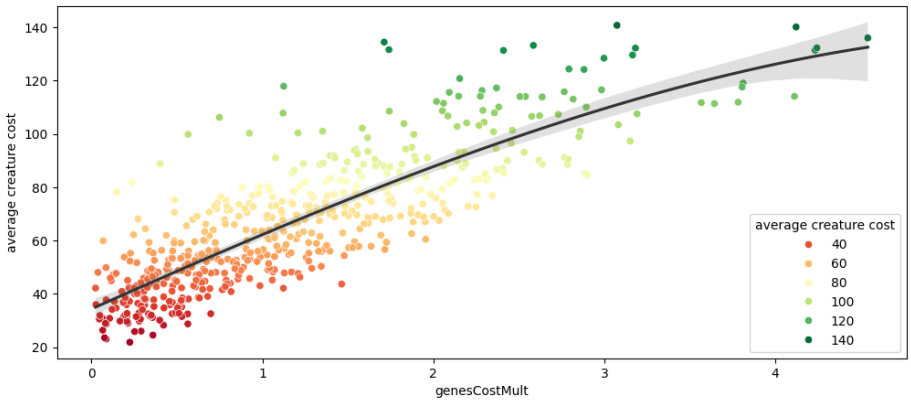
Rysunek 13: Wykres szansy mutacji części ciała, a średniego kosztu komórki.

Jak widać na wykresie, występuje pewna korelacja. Nie jest ona silna. Niepewność również wzrasta, do bardzo wysokiego poziomu.



Rysunek 14: Wykres mnożnika kosztu części ciała, a średniego kosztu komórki.

Jak widać na wykresie, średni koszt komórki wzrasta razem z mnożnikiem kosztu części ciała, aż do czasu. Potem zaczyna się obniżać. Być może to przez to, że gdy koszt części ciała jest tak wysoki, komórki wymierają, co może sprawiać, że przeżyją tylko te, które mogą reprodukować się najniższym kosztem.



Rysunek 15: Wykres mnożnika kosztu genów, a średniego kosztu komórki.

Korelacja jest bardzo silna i liniowa. Dopiero przy najwyższych wartościach mnoznika kosztu genów koszt komórek zaczyna wolniej rosnąć, co może być spowodowane ewolucją i przetrwaniem tylko tych, którzy mogą wydać potomstwo z mniejszą inwestycją energetyczną.

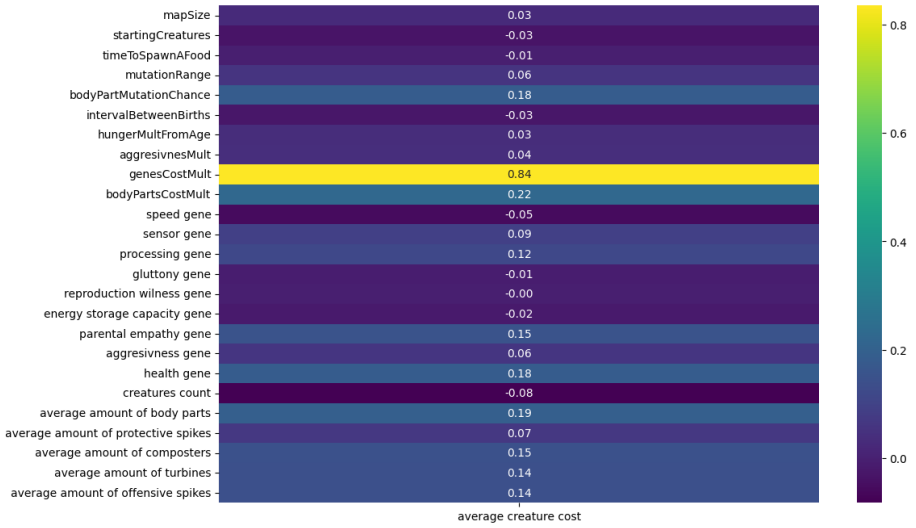
5.4 Wnioski z estymacji średniego kosztu komórki z początkowych parametrów

Okazuje się, że występuje bardzo silna korelacja między średnim kosztem komórki, a mnoznikiem kosztu genów. Widoczna jest również korelacja między szansą mutacji części ciała czy mnoznikiem kosztu części ciała. Wynik regresji na poziomie 0,8 jest wysoki, biorąc pod uwagę ilość losowości w symulacji. Nie wygląda na to, aby doszło do nadmiernego dopasowania (overfittingu) - wyniki estymatorów dla danych treningowych są równie wysokie.

6 Regresja liniowa na podstawie rezultatów i parametrów początkowych

Skoro mamy dane pozostałych rezultatów, interesujące byłoby sprawdzić, jak dołączenie tych danych wpłynęłoby na estymację kosztu komórek.

6.1 Kolumna korelacji



Rysunek 16: Kolumna korelacji średniego kosztu komórki od parametrów początkowych symulacji i rezultatów.

Ponieważ wartość korelacji jest niska dla wielu kolumn, estymacja powstanie na bazie tylko niektórych parametrów początkowych i rezultatów. Do poprzednich parametrów (5.1) dodane zostały:

- średnia wartość genu szybkości
- średnia wartość genu sensorycznego
- średnia wartość genu przetwarzania jedzenia
- średnia wartość genu łakomstwa
- średnia wartość genu empatii rodzicielskiej
- średnia wartość genu agresji
- średnia wartość genu zdrowia
- średnia ilość części ciała
- średnia ilość kolców ochronnych
- średnia ilość kompostowników
- średnia ilość turbin
- średnia ilość kolców ofensywnych

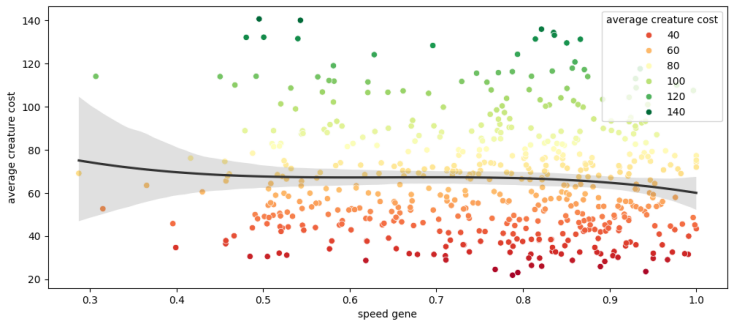
6.2 Wynik regresji liniowej

Skalując dane za pomocą StandardScaler, oto wyniki regresji:

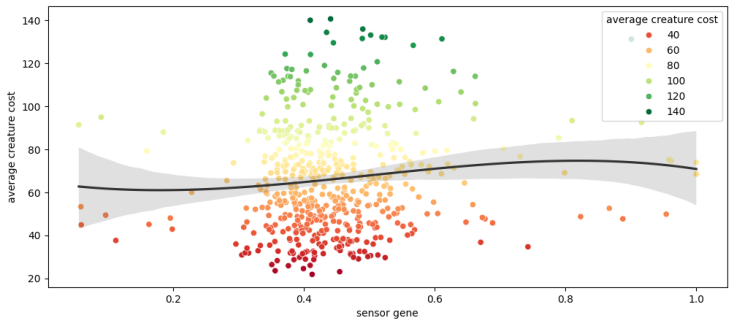
Regression Score on:	R^2
Test data	0.9023
Train data	0.9124
Full data	0.9103

6.3 Wykresy wartości parametrów początkowych i rezultatów, a średniego kosztu komórki

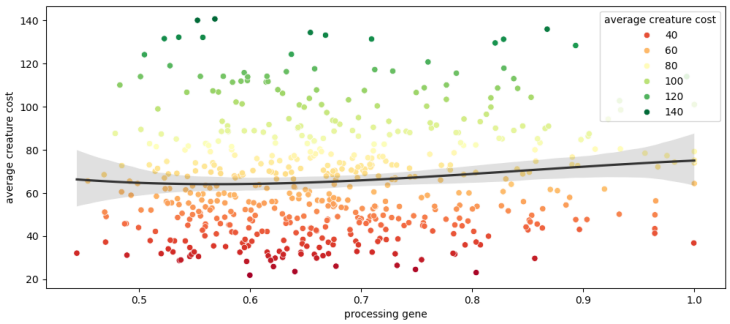
3 wykresy pojawiły się już w poprzednim dziale: 14 15 13



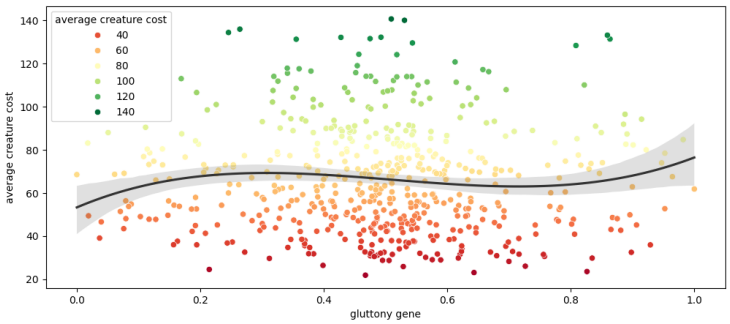
Rysunek 17: Wykres średniej wartości genu szybkości, a średniego kosztu komórki.



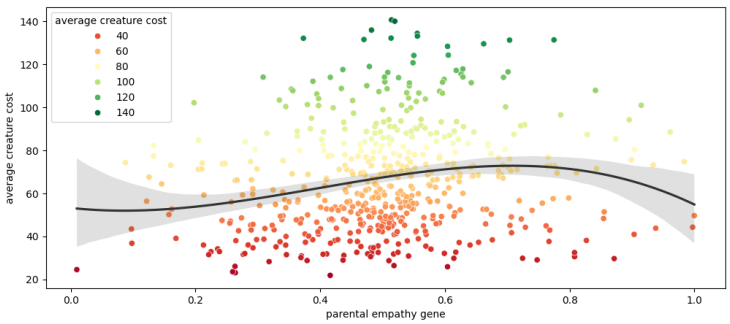
Rysunek 18: Wykres średniej wartości genu sensorycznego, a średniego kosztu komórki.



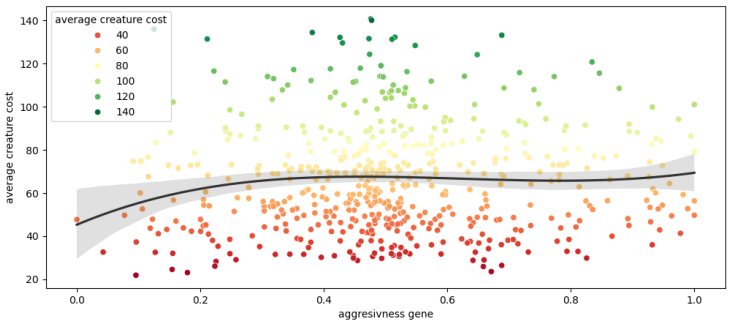
Rysunek 19: Wykres średniej wartości genu przetwarzania jedzenia, a średniego kosztu komórki.



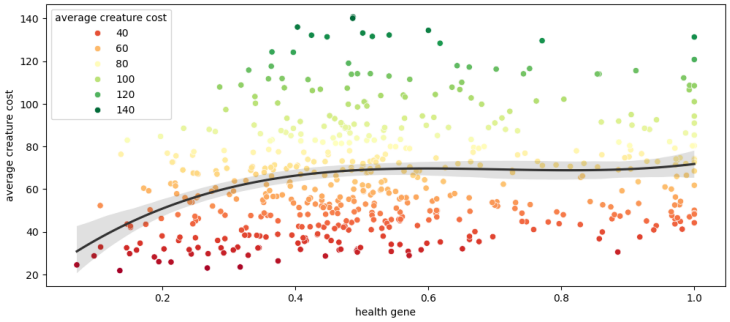
Rysunek 20: Wykres średniej wartości genu łakomstwa, a średniego kosztu komórki.



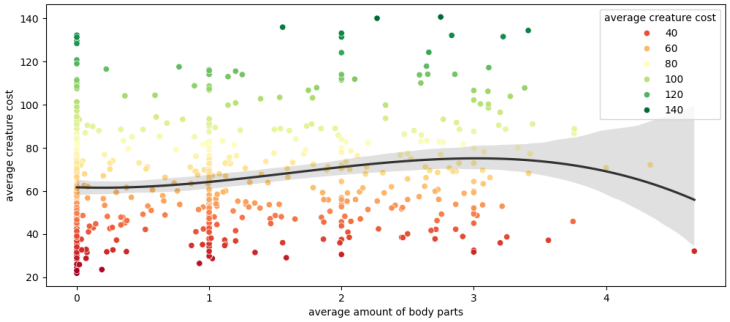
Rysunek 21: Wykres średniej wartości genu empatii rodzicielskiej, a średniego kosztu komórki.



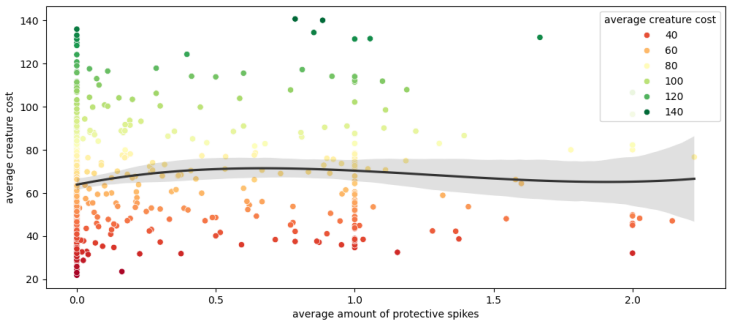
Rysunek 22: Wykres średniej wartości genu agresji, a średniego kosztu komórki.



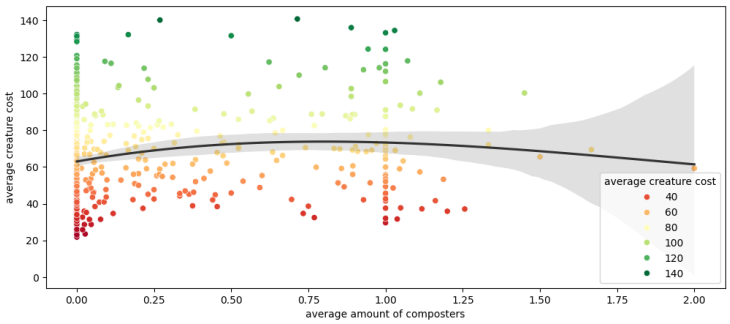
Rysunek 23: Wykres genu zdrowia, a średniego kosztu komórki.



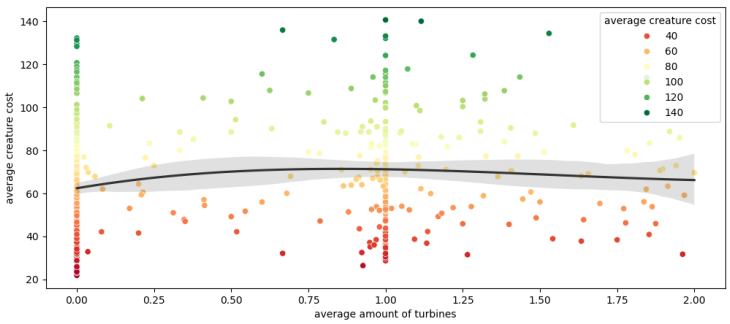
Rysunek 24: Wykres średniej ilości części ciała, a średniego kosztu komórki.



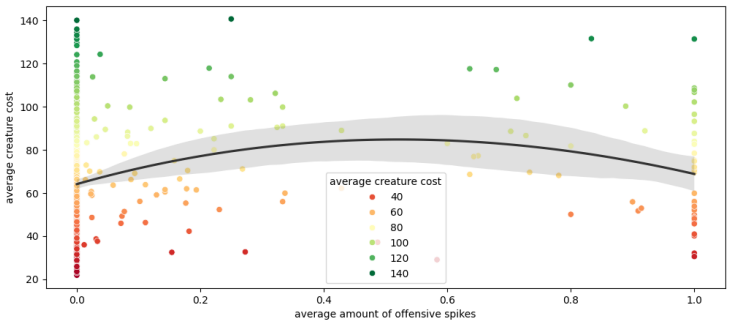
Rysunek 25: Wykres średniej ilości kolców ochronnych, a średniego kosztu komórki.



Rysunek 26: Wykres średniej ilości kompostowników, a średniego kosztu komórki.



Rysunek 27: Wykres średniej ilości turbin, a średniego kosztu komórki.



Rysunek 28: Wykres średniej ilości kolców ofensywnych, a średniego kosztu komórki.

Patrząc na wykresy średnich ilości ciała, widać anomalie wśród danych - wiele tych średnich wartości to liczby całkowite. Jest to jednak oczekiwane, gdyż pod koniec symulacji może się zdarzyć, że pozostanie niewielka liczba komórek, wszystkie mające tą samą ilość części ciała.

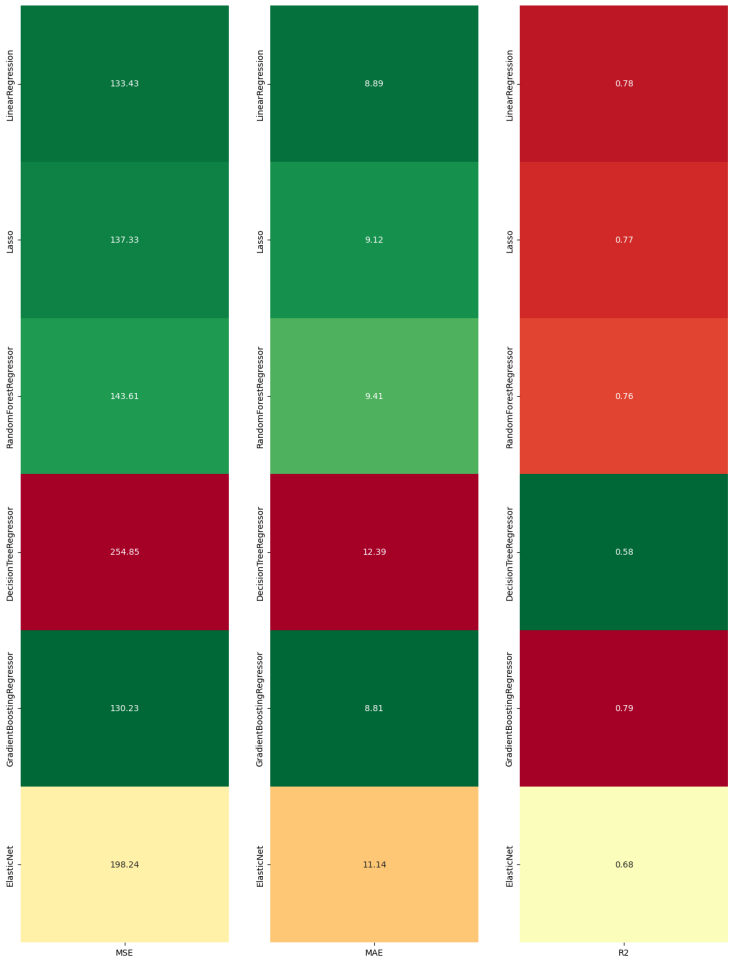
6.4 Wnioski z estymacji średniego kosztu komórki z rezultatów i początkowych parametrów

Wysokość korelacji uległa poprawie w porównaniu z estymacją z samych rezultatów, z poziomu 0,8 do około 0,9. Jest to wysoki wynik, biorąc pod uwagę ilość losowości w symulacji. Nadal widać jednak, że to sam mnożnik kosztu genów odpowiada za dużą część tego wyniku. Ilość analizowanych danych również się mocno zwiększyła. Nie wygląda na to, aby doszło do nadmiernego dopasowania (overfittingu) – wyniki estymatorów dla danych treningowych są równie wysokie.

7 Porównanie estymacji wśród różnych modeli

Ostatnim etapem było ponowna estymacja kosztu komórek na podstawie samych parametrów początkowych, ale z użyciem różnych modeli. Wybrane zostały trzy, które osiągnęły najlepsze wyniki.

7.1 Testowanie modeli



Rysunek 29: Wartości MSE, MAE i R^2 dla przetestowanych modeli.

Na danych z działu Wynik regresji liniowej przetestowano następujące modele: LinearRegression, Lasso, RandomForestRegressor, DecisionTreeRegressor, GradientBoostingRegressor i ElasticNet. Stworzona została heatmapa, zawierająca miary MSE, MAE i R^2 przetestowanych modeli. Po przeanalizowaniu wyników, do analizy wybrano LinearRegression, RandomForestRegressor i GradientBoostingRegressor.

7.2 Szukanie najlepszych parametrów modeli

Każdy model przyjmuje inne parametry, decydujące o poziomie jego dopasowania do danych.

7.2.1 Linear Regression

Regresja liniowa z cechami wielomianowymi [2] to technika uczenia maszynowego, która rozszerza możliwości tradycyjnej regresji liniowej poprzez uwzględnienie wielomianowych przekształceń cech wejściowych. Dzięki temu model może uchwycić nieliniowe zależności między zmiennymi niezależnymi a zmienną zależną, co prowadzi do bardziej precyzyjnych prognoz w przypadku nieliniowych danych.

Oto niektóre z najważniejszcyh parametrów lasu losowego:

- degree: Stopień wielomianu używany do transformacji cech. Wyższy stopień może lepiej dopasować się do złożonych zależności, ale zwiększa ryzyko nadmiernego dopasowania.
- include_bias: Wartość logiczna określająca, czy dodać kolumnę jednostkową (bias) do przekształconych cech. Zwykle dodanie kolumny bias poprawia jakość modelu.

7.2.2 Random Forest

Las losowy [3] to złożony model uczenia maszynowego, który łączy wiele drzew decyzyjnych, aby poprawić dokładność i kontrolować nadmierne dopasowanie. Każde drzewo w lesie losowym jest trenowane na losowym podzbiorze danych z użyciem losowych podzbiorów cech, co zwiększa różnorodność i ogólną wydajność modelu.

Oto niektóre z najważniejszcyh parametrów lasu losowego:

- n_estimators: Liczba drzew w lesie. Większa liczba drzew zwykle poprawia dokładność modelu, ale zwiększa również czas obliczeń.
- min_samples_split: Minimalna liczba próbek wymagana do podziału węzła. Większe wartości pomagają w uniknięciu nadmiernego dopasowania.
- min_samples_leaf: Minimalna liczba próbek, które muszą znajdować się w liściu. Wyższe wartości mogą poprawić ogólną stabilność modelu.
- max_features: Liczba cech rozważanych przy każdym podziale. Może być wartością absolutną, procentową lub "sqrt"(pierwiastek kwadratowy liczby cech) lub "log2".

7.2.3 Gradient Boosting

Gradient Boosting [1] to zaawansowana technika uczenia maszynowego, która iteracyjnie łączy słabe modele, takie jak małe drzewa decyzyjne, aby tworzyć

silny model. Każde kolejne drzewo koryguje błędy popełnione przez poprzednie drzewa, co prowadzi do coraz lepszej wydajności modelu.

Oto niektóre z najważniejszych parametrów lasu losowego:

- n_estimators: Liczba drzew w sekwencji. Większa liczba estymatorów może poprawić dokładność, ale zwiększa również ryzyko nadmiernego dopasowania.
- min_samples_split: Minimalna liczba próbek wymagana do podziału węzła. Większe wartości pomagają w uniknięciu nadmiernego dopasowania.
- min_samples_leaf: Minimalna liczba próbek, które muszą znajdować się w liściu. Wyższe wartości mogą poprawić ogólną stabilność modelu.
- max_features: Liczba cech rozważanych przy każdym podziale. Może być wartością absolutną, procentową lub "sqrt"(pierwiastek kwadratowy liczby cech) lub "log2".
- learning_rate: Szybkość uczenia, która skaluje wkład każdego drzewa. Mniejsza wartość learning_rate wymaga większej liczby estymatorów, aby osiągnąć ten sam poziom wydajności.

7.3 Dopasowanie modeli

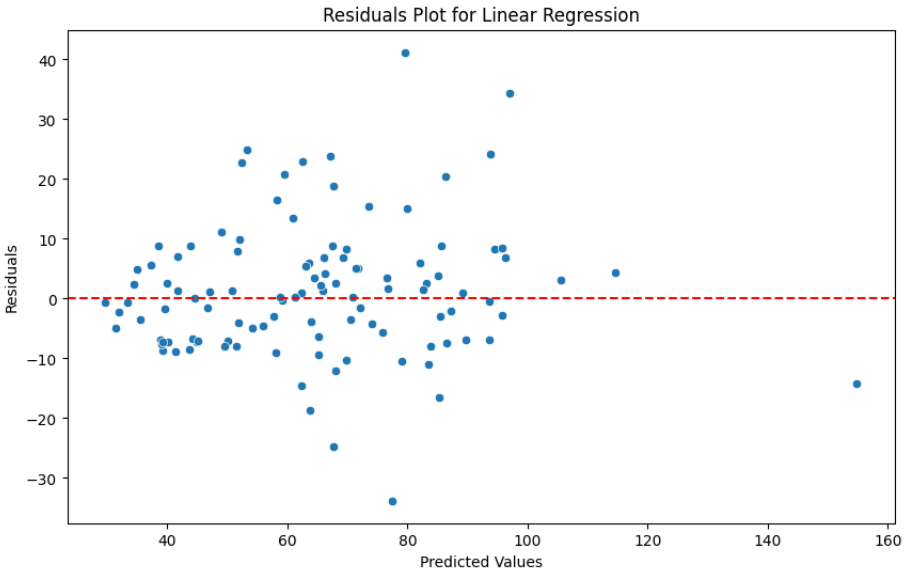
Używając wybranych modeli wraz z siatkami parametrów, przetestowana została każda kombinacja podawanych niżej parametrów.

- Linear Regression
  - degrees: [1, 2, 3]
  - include_bias = [True, False]
- Random Forest
  - n_estimators: [100, 200, 300],
  - max_depth: [None, 5, 10],
  - min_samples_split: [2, 5, 10]
- Gradient Boosting
  - n_estimators = [125, 150, 200, 300]
  - min_samples_splits = [2, 5, 10]
  - min_samples_leafs = [1, 3, 5]
  - max_features = [0.2, 'sqrt', 'log2', 1]
  - learning_rates = [0.05, 0.075, 0.1, 0.15]

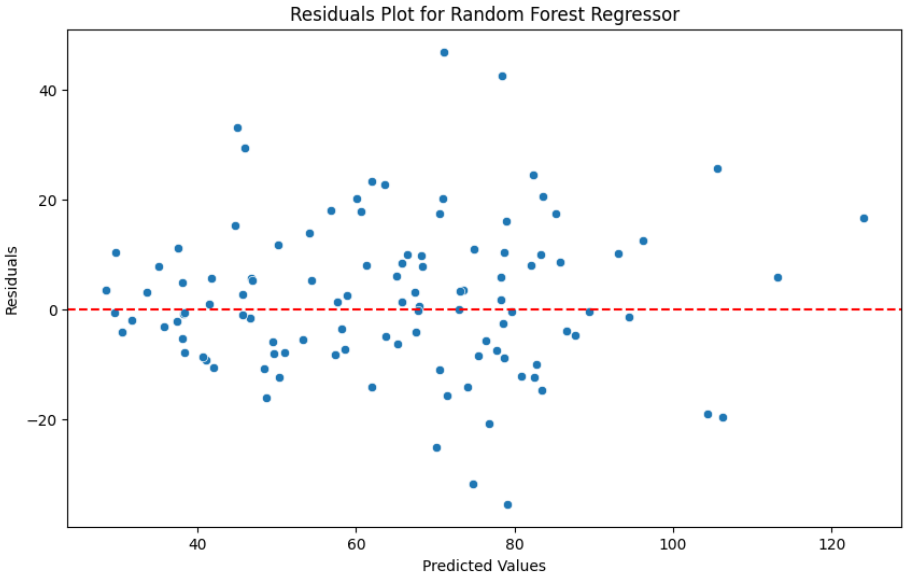
Oto parametry, które dla danego modelu pozwalały estymować najlepiej:

- Linear Regression - {degree: 2, include_bias: True}
- Random Forest Regressor - {n_estimators: 300, min_samples_split: 5, min_samples_leaf: 1, max_features: 0.2}
- GradientBoostingRegressor - {n_estimators: 125, min_samples_split: 5, min_samples_leaf: 1, max_features: 0.2, learning_rate: 0.075}

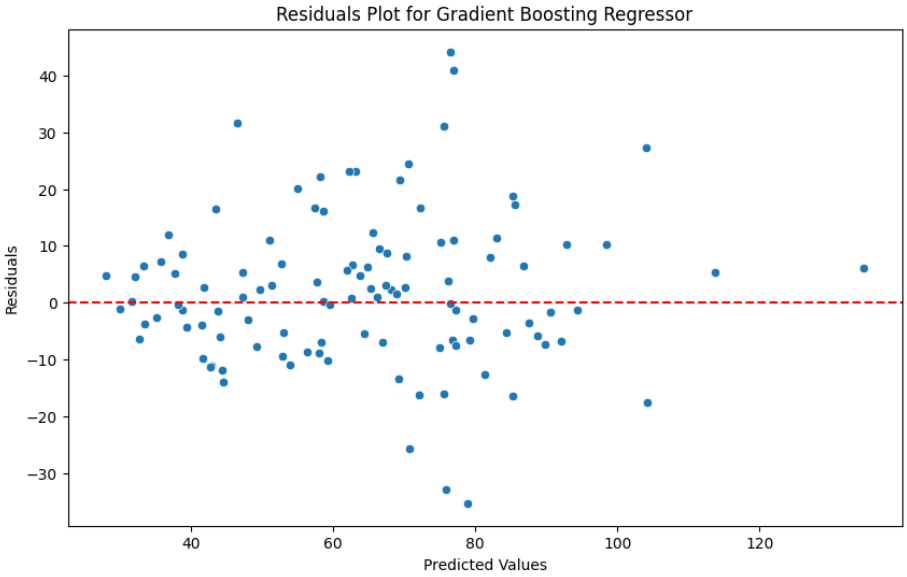
7.4 Wykresy reszt



Rysunek 30: Wykres reszt dla Linear Regression



Rysunek 31: Wykres reszt dla Random Forest Regressor



Rysunek 32: Wykres reszt dla Gradient Boosting Regressor

Wykresy reszt (residuals plot) są dość podobne dla wszystkich modeli. Są jednak pewne różnice. Wygląda na to, że wartości przewidywane metodą Linear

Regression są niższe, niż w przypadku innych modeli.

7.5 Wyniki

Tabela 4: Wyniki różnych modeli regresji

Model	**MSE**	**MAE**	$\mathbf{R^2}$
Linear Regression	126.615	8.188	0.788
Random Forest Regressor	165.674	9.947	0.722
Gradient Boosting Regressor	149.676	9.320	0.749

Tabela 5: Wyniki różnych modeli regresji na zbiorach testowych i treningowych

Model	$\mathbf{R^2}$ **(Test)**	$\mathbf{R^2}$ **(Train)**
Linear Regression	0.788	0.833
Random Forest Regressor	0.722	0.939
Gradient Boosting Regressor	0.749	0.918

7.6 Wnioski

Najwyższy wynik wśród wartości testowych osiągnął model Linear Regression. Modele Random Forest Regressor i Gradient Boosting Regressor osiągnęły bardzo wysokie wyniki dla danych treningowych, ale zauważalnie niższe dla danych testowych, co sugeruje wystąpienie zjawiska overfitting. Wynik zoptymalizowanego modelu Linear Regression jest praktycznie taki sam, jak w 5.2. Wyniki zbliżone do 0,8 można określić jako dobre, biorąc pod uwagę losowość i złożoność symulacji.

7.7 Uwzględnienie wszystkich parametrów początkowych

Na koniec został przeprowadzony dodatkowy eksperyment. Jak na wyniki estymacji wpłynęłoby uwzględnienie wszystkich parametrów początkowych? Oto wyniki:

Tabela 6: Wyniki różnych modeli regresji na zbiorach testowych i treningowych z uwzględnieniem wszystkich parametrów początkowych.

Model	$\mathbf{R^2}$ **(Test)**	$\mathbf{R^2}$ **(Train)**
Linear Regression	0.818	0.879
Random Forest Regressor	0.751	0.841
Gradient Boosting Regressor	0.811	0.987

Gradient Boosting Regressor doświadczył mocnego overfittingu. Ale zarówno ten model, jak i Linear Regression osiągnęły wyższe wyniki R^2 dla danych testowych, niż z uwzględnieniem tylko wybranych parametrów początkowych. Wyniki są zauważalnie wyższe.

8 Wnioski ogólne

Oczywistym wnioskiem jest to, że dużą częścią niezłych wyników estymacji były parametry początkowe "mnożnik kosztu genów"i "mnożnik kosztu części ciała". Są to parametry, które mogły w złożony sposób wpłynąć na symulację, wprowadzając nieoczekiwane zależności. Stało się tak tylko w części, a zależności między tymi parametrami a średnim kosztem komórki były dość silne i dość liniowe. Na pewno ciekawym eksperymentem byłoby przeprowadzenie analizy jeszcze raz, ale utrzymując obe te parametry jako stałe.

Kolejnym problemem mógł być zbyt krótki czas symulacji. Prawdopodobnie przy dłuższym czasie symulacji wyniki mogłyby być interesujące. Ilość przeprowadzonych symulacji wydaje się dość adekwatna - overfitting występuje, ale w większości przypadków nie jest bardzo silny.

Ta analiza na pewno pomogłaby podczas rozwoju gry i mogłaby być świetnym narzędziem do optymalizacji parametrów wewnątrz kodu.

Literatura

[1] Gradient boosting. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html. Accessed: 2024-06-11.

[2] Polynomial features. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PolynomialFeatures.html. Accessed: 2024-06-11.

[3] Random forest. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html. Accessed: 2024-06-11.